



Scientific Computing with (Short version)

Rafael Palacios (Feb/2014)

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Introduction to Matlab

- ¿What is Matlab?
 - Matlab = **Matrix Laboratory**.
 - Interactive program for mathematical operations and graphical representation
 - Company: The Mathworks Inc (Natick, MA).
<http://www.mathworks.com>
 - Created in California by Jack Little and Cleve Moler in 1984, for matrix computation without the need of background in programming (no loops).

Matlab 7 development environment

The screenshot displays the Matlab 7 development environment interface. The main window is titled "Figuras - Figure 1" and contains a 3D surface plot of the "peaks" function. The plot shows a complex surface with multiple peaks and valleys, colored with a gradient from blue (low values) to red (high values). The axes are labeled x, y, and z. The z-axis ranges from -5 to 5, the x-axis from -3 to 3, and the y-axis from -2 to 2.

On the left side, the "Workspace" window shows a table with the following content:

Name	Value
x	<20x20 double>

Below the Workspace is the "Command History" window, which lists the following commands:

```
txt=nombres  
matriculas  
whos  
matriculas  
whos  
i  
  ymed(12)  
  xmed(12)  
%-- 11/13/04 8:28 AM --  
  ver  
%-- 11/13/04 8:30 AM --  
  bench  
%-- 11/26/04 4:38 PM --  
  peaks  
  x=rand(20);  
  plot(x(4,1:20), 'Disp  
  peaks
```

The "Command Window" at the bottom right shows the following code and output:

```
>> x=rand(20);  
>> plot(x(4,1:20), 'DisplayName', 'x(4,1:20)', 'YDataSource', 'x(4,1:20)');  
>> plot(x(4,1:20), 'DisplayName', 'x(4,1:20)', 'YDataSource', 'x(4,1:20)');  
>> peaks  
  
z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...  
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...  
    - 1/3*exp(-(x+1).^2 - y.^2)
```

Arrows point from the labels "Workspace", "command history", "Figuras", and "command window" to their respective components in the screenshot.

Algunas mejoras de Matlab 7

- Immediate data plot

The screenshot shows the MATLAB interface with a data table. A context menu is open over the table, highlighting the 'Plot selected columns' option. The table contains numerical data in 8 columns and 22 rows. The menu options include standard editing actions (Cut, Copy, Paste, etc.), 'Insert...', 'Delete...', 'Clear Contents', 'Create Variable from Selection', 'Plot selected columns', and several plotting functions: 'bar (x(4:17,2:3))', 'imagesc (x(4:17,2:3))', 'contour (x(4:17,2:3))', 'surf (x(4:17,2:3))', 'mesh (x(4:17,2:3))', and 'More Plots...'. The 'Plot selected columns' option is highlighted in blue.

	1	2	3	4	5	6	7	8
1	0.95013	0.057891	0.83812	0.49655	0.79482	0.58279	0.41537	0.68330
2	0.23114	0.35287	0.01964	0.89977	0.95684	0.4235	0.305	0.21256
3	0.60684	0.81317	0.68128	0.82163	0.52259	0.51551	0.87437	0.83924
4	0.48598	0.0098613	0.37081	0.81484	0.89844	0.22225	0.015009	0.62876
5	0.8913	0.13889	0.8				0.76795	0.13370
6	0.7621	0.20277	0.50				0.97084	0.20710
7	0.45647	0.19872	0.70				0.99008	0.60700
8	0.018504	0.60379	0.42				0.78886	0.62980
9	0.82141	0.27219	0.30				0.43866	0.37040
10	0.4447	0.19881	0.18				0.49831	0.57510
11	0.61543	0.015274	0.19				0.21396	0.45140
12	0.79194	0.74679	0.68				0.64349	0.043890
13	0.92181	0.4451	0.30				0.32004	0.027180
14	0.73821	0.93181	0.54				0.9601	0.31260
15	0.17627	0.46599	0.15				0.72663	0.012860
16	0.40571	0.41865	0.6				0.41195	0.38390
17	0.93547	0.84622	0.37				0.74457	0.68310
18	0.9169	0.52515	0.88				0.26795	0.092840
19	0.41027	0.20265	0.85				0.43992	0.035330
20	0.89365	0.67214	0.59				0.93338	0.61200
21								
22								

Tamaño máximo
de la matriz:
524288 elementos

Development environment

The screenshot displays the MATLAB development environment interface. The top menu bar includes File, Edit, Debug, Desktop, Window, and Help. The current directory is set to C:\MATLAB701\work. The workspace window on the left shows variables: ans (value [1.162 1.532 0.56...]), i (value 20), x (value <3x5 double>), and z (value <6x2 double>). The command window on the right shows the following commands and output:

```
?? x=zeros(6,2)
Error: Unbalanced or misused parentheses or brackets.

>> x=ones(3,5)
x =
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1

>> z=zeros(6,2)
z =
    0    0
    0    0
    0    0
    0    0
    0    0
    0    0

>> x=ones(3,5)
x =
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1

>>
```

The Command History window at the bottom left shows a list of executed commands, including help textscan, bench, help bench, saveworkspace, save workspace, save kk x, i, save kk x i, x=ones(3), x=ones(3,5), x=zeros(6,2), x=ones(3,5), z=zeros(6,2), and x=ones(3,5).

Command window

Basic commands

- `ver` → version number, license code and toolbox versions
 - License 46431: Research
 - License 205966: Only for teaching
- `whos` → list of variables
- `save archivo` → save all variables
- `save archivo a b` → save variables a & b
- `load archivo` → load file and create variables
- `quit` → exit

Editor

- Matlab includes and editor for writing programs and functions

Ejecución por secciones en cell mode

Controles del debugger

sintaxis

```
1 function [med,des]=med_des(x)
2 % Funciona para calcular la media y la desviación a la vez
3 % [med,des]=med_des(x)
4 %
5 % Rafael Palacios (nov/2004)
6 - med=mean(x(:));
7 - des=std(x(:));
8 |
```

Documentation

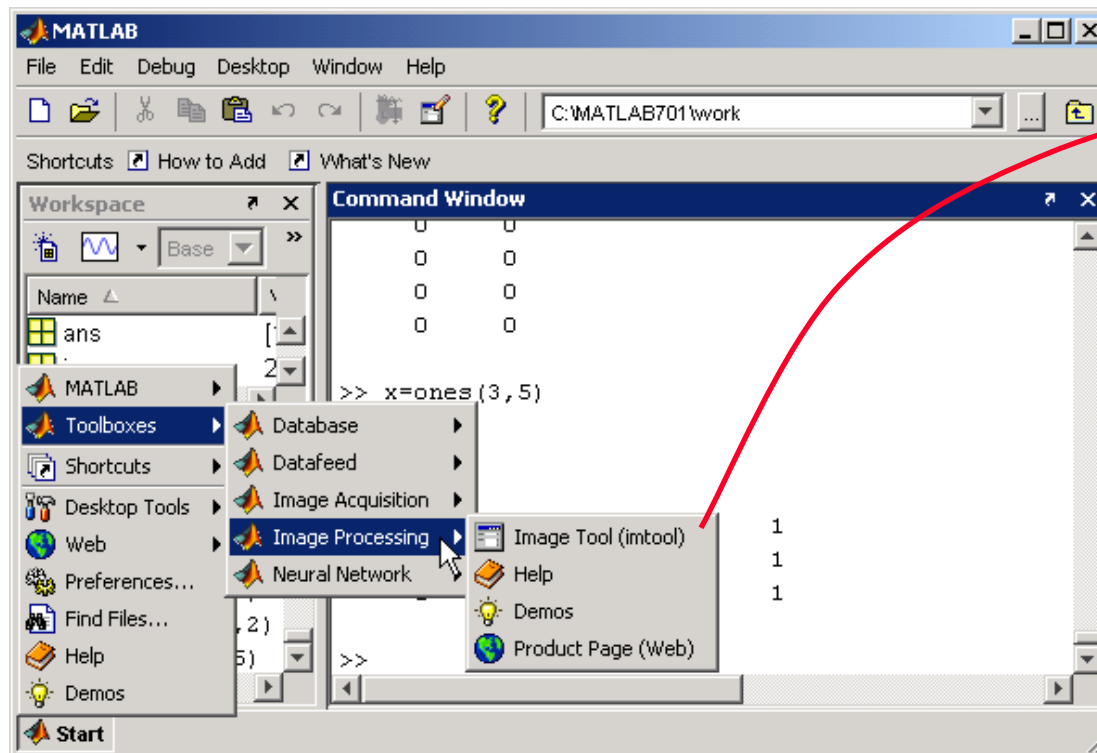
- Ayuda en modo texto mediante comandos
 - help función → muestra la ayuda de una función
 - help vale tanto para las funciones del sistema como para desarrollos propios
- Ayuda de tipo gráfico

The image shows two screenshots from the MATLAB software. The left screenshot shows the MATLAB desktop environment with the 'Help' menu item highlighted in the 'Start' menu. A red arrow points from this menu item to the right screenshot. The right screenshot is titled 'Hypertext Help Window' and displays the MATLAB help browser interface. It features a 'Help Navigator' on the left with a tree view of documentation topics, including 'MATLAB', 'MATLAB Compiler', 'MATLAB Report Generator', 'Database Toolbox', 'Datafeed Toolbox', 'Image Acquisition Toolbox', 'Image Processing Toolbox', 'Neural Network Toolbox', and 'Support and Web Services'. The main content area shows the 'MATLAB®' page with a title bar 'Title: MATLAB®' and a table of values. Below the table are sections for 'Functions:' (with links for 'By Category' and 'In Alphabetical Order'), 'Handle Graphics:' (with link for 'Object Properties'), 'Documentation Set' (with links for 'Getting Started', 'User Guides', 'Programming Tips', and 'Examples in Documentation'), 'Product Demos' (with link for 'MATLAB Demos'), and 'What's New'.

Start / Help

Toolboxes

- Specific libraries for different scientific topics. Include:
 - User's Guide [HTML, PDF]
 - Reference Guide [HTML, PDF]
 - Demo Programs
 - Demo applications (called tool) ready to use



Try it yourself

- Everybody
 - Open Matlab
 - Identify all sections of the environment
 - Get the list of toolboxes installed
 - Declare a matrix, ex: `a(3,4)=25;`
 - Edit values of the matrix
 - Plot columns/rows
- Advanced
 - Plot surface. Edit surface.
 - Use a demo program of one toolbox

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Variables

- Matlab doesn't require to declare variable or to specify the size of arrays
 - Variables are auto-declared when assigned
 - Memory is reallocated automatically

```
>> x=5;
>> y=20;
>> z=x*y

z =

    100

>> a=load('data.txt');
>> st='ho1a';
```

using ';' the value is assigned but the result is not displayed

without ';' the final result is shown

Vectors and Matrices

- Matlab considers that all variables are matrices
- Vectors and scalars are special cases (size 1).

Some examples on how to initialize **row vectors**

```
>> x=[1,2,3,5,7,11,13]; → [ 1 2 3 5 7 11 13 ]
>> x=[1 2 3 5 7 11 13]; → [ 1 2 3 5 7 11 13 ]

>> y=1:5; → [ 1 2 3 4 5 ]
>> even=2:2:10; → [ 2 4 6 8 10 ]
>> odd_down=9:-2:1; → [ 9 7 5 3 1 ]

>>a(5)=7; → [ 0 0 0 0 7 ]
```

Vectors and Matrices

Examples on how to initialize matrices

```
>> M = [1 2 3; 4 5 6; 7 8 9];
```

1	2	3
4	5	6
7	8	9

```
>> ceros=zeros(2,5);
```

0	0	0	0	0
0	0	0	0	0

```
>> unos=ones(3,4);
```

1	1	1	1
1	1	1	1
1	1	1	1

```
>> M2=[ 20, 21, 22; M];
```

```
>> M2=[[20, 21, 22]; M];
```

```
>> M3=[ [15;16;17], M];
```

20	21	22
1	2	3
4	5	6
7	8	9

```
>> aleatorio=rand(20,30);
```

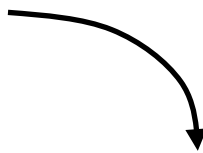
```
>> normal=randn(20,30);
```

15	1	2	3
16	4	5	6
17	7	8	9

Accessing elements of a matrix

- Matlab uses parenthesis () to access elements of a matrix
- Indexes start at 1, so the first of matrix mat is `mat(1,1)`
- Example: `a(3,5)=56.8;`

0.1737	0.3421	0.6391	0.1632	0.2313
0.7858	0.7742	0.0934	0.2763	0.8453
0.3656	0.1478	0.9288	0.1310	0.7264
0.7769	0.1482	0.4851	0.0232	0.6947




0.1737	0.3421	0.6391	0.1632	0.2313
0.7858	0.7742	0.0934	0.2763	0.8453
0.3656	0.1478	0.9288	0.1310	56.8000
0.7769	0.1482	0.4851	0.0232	0.6947

Accessing elements of a matrix

- Ejemplo 2: `a([2,3],[2,4])=ones(2,2);`
o bien: `a([2,3],[2,4])=1;`

0.1737	0.3421	0.6391	0.1632	0.2313
0	0	0	0	0.8453
0	0	0	0	56.8000
0.7769	0.1482	0.4851	0.0232	0.6947



0.1737	0.3421	0.6391	0.1632	0.2313
0	1.0000	0	1.0000	0.8453
0	1.0000	0	1.0000	56.8000
0.7769	0.1482	0.4851	0.0232	0.6947

Accessing elements of a matrix

- The symbol ' : ' means “all elements”
- It may be used as “all elements in row” or “all elements in the matrix”, etc.

0.1737	0.3421	0.6391	0.1632	0.2313
0.7858	0.7742	0.0934	0.2763	0.8453
0.3656	0.1478	0.9288	0.1310	56.8000
0.7769	0.1482	0.4851	0.0232	0.6947

→ `a(3,:)`
`size(a(3,:))` → [1 5]

↓
`a(:,2)`
`size(a(:,2))` → [4 1]

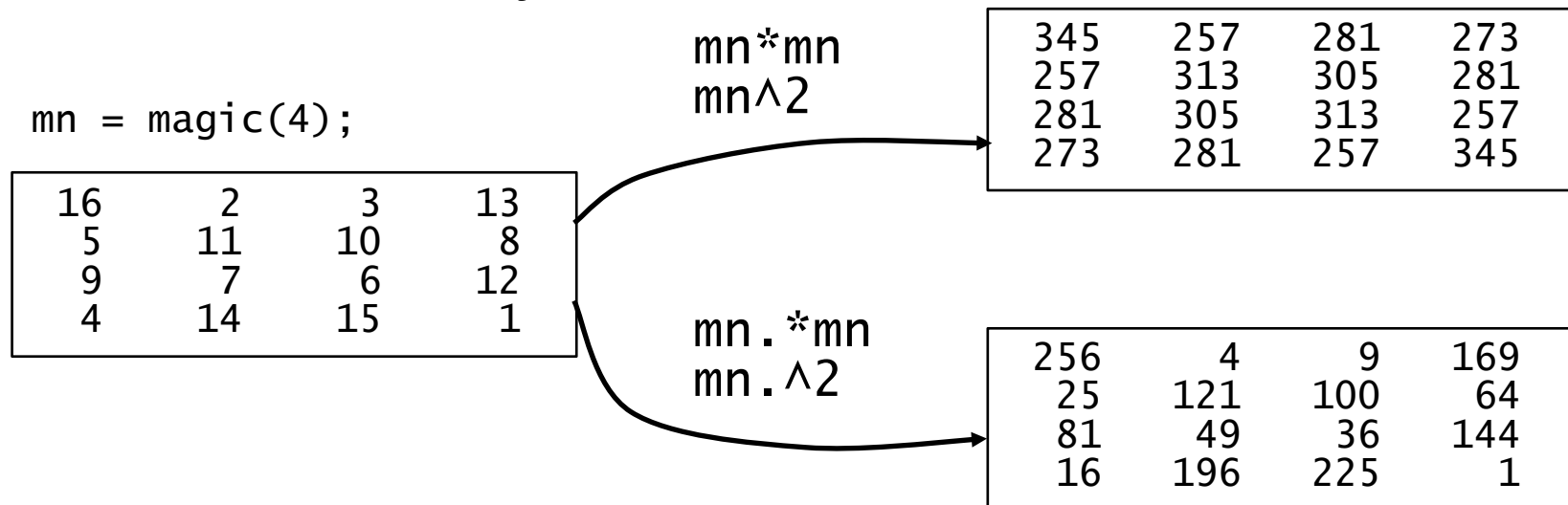
↘
`a(:)` → all elements
`size(a(:))` → [20 1]
returns a column vector

Basic mathematical operators

- Arithmetic operators: + - * / ^
 - Matlab works with matrices, in contrast with other programming languages that only work with scalar values

```
>> a=rand(2,5);  
>> b=rand(5,2);  
>> c=a*b;           % matrix 2x2  
>> d=b*a;           % matrix 5x5
```

- Operations element-by-element: + - .* ./ .^



Other matrix operations

- Sum: `sum`

```
>> b=sum(A);    % if A is a matrix, add elements by column. b is a row vector
>> c=sum(b);    % if b es vector, all elements are added. c is a scalar

>> c=sum(sum(a)); % addition of all elements in a
>> c=sum(a(:));  % addition of all elements in a
```

- Mean and stander deviation: `m=mean(A);`
`sigma=std(A);`
- Elements of the diagonal: `v=diag(A);`
- Left division: `x=A\B;` The least squares solution for $Ax = b$ is obtained by means of $x = A \setminus b$;
- Determinant: `c=det(A);`
- Inverse: `B=inv(A);`
- Eigenvalues: `v=ein(A);`
- Absolute values, or complex module: `B=abs(A);`

Data types

- Matlab uses type **double** according to the IEEE standard
- It can manage special values like inf (infinity) y NaN (not-a-number)
- Complex number are used automatically when needed.

```
>> a=123/0
Warning: Divide by zero.
a =
    Inf

>> b=0/0
Warning: Divide by zero.
b =
    NaN

>> Inf-Inf
ans =
    NaN

>> c=15+sqrt(-1)
c =
 15.0000 + 1.0000i
```

Try it yourself

- Everybody
 - Try different examples to access data and extract data
- Advanced
 - Build a vector containing NaN values. Plot the vector
 - Set axis limit to $-\text{inf}$ or $+\text{inf}$

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Scripts

- One **script** is a sequence of instructions that matlab can save in a `.m` file

```
%Script de ejemplo

%% Inicio
a=magic(4);
fprintf('Inicio cálculos\n');

%% Traza
traza=sum(diag(a));

%% Resultado
fprintf('La traza vale: %f\n',traza)
```

`ejem_script.m`

- It is executed by the filename:

```
>> ejem_script
```

Functions (Call)

- Matlab functions can receive several arguments and they can also return several results:

```
[m,d]=med_des(x);
```

- Functions can have optional arguments

```
mit=imread('cameraman.tif','TIFF');  
mit=imread('cameraman.tif');
```

- It is not necessary to assign all results

```
[mit,map]=imread('imageman.gif');  
mit=imread('imageman.gif');
```

Functions (declaration)

- Functions are also written in .m files that must be located in the current working directory (or a directory specified in the path)

```
function [med,des]=med_des(x)
% Funciona para calcular la media y la desviación a la vez
% [med,des]=med_des(x)
%
% Rafael Palacios (nov/2004)
med=mean(x(:));
des=std(x(:));
```

Med_des.m

This is the information that you get if you type: `help med_des`

Logical expressions

- Relational operators: `~=` `==` `>` `<` `>=` `<=`
- Logical operators:
 - `&&` Short-circuit AND
 - `||` Short-circuit OR
 - `&` AND
 - `|` OR
- There is a function for `xor`, but not an operator

Control de Flujo: if

- if

```
if a > b
    tmp=a;
    a=b;
    b=tmp;
end
```

```
if rem(n,2) ~= 0
    M = odd_magic(n)
elseif rem(n,4) ~= 0
    M = single_even_magic(n)
else
    M = double_even_magic(n)
end
```

In contrast with C, Matlab does not require parenthesis for the logical expression

Control de Flujo: for

- for loop

```
for n = 3:32
    r(n) = rank(magic(n));
end
```

```
a=[];
for n = [ 1 2 3 5 7 11 ]
    a = [a, isprime(n)];
end
```

Control de Flujo: while

- while loop

```
while ~isprime(x)
    x = x + 1;
end
```

Control de Flujo: switch

- switch-case

```
switch (rem(n,4)==0)+(rem(n,2)==0)
    case 0
        M = odd_magic(n)
    case 1
        M = single_even_magic(n)
    case 2
        M = double_even_magic(n)
    otherwise
        error('This is impossible')
end
```

In contrast with C, Matlab doesn't use break.

Control de Flujo: try

- try-catch

```
try
    statement
    ...
    statement
catch
    statement
    ...
    statement
end
```

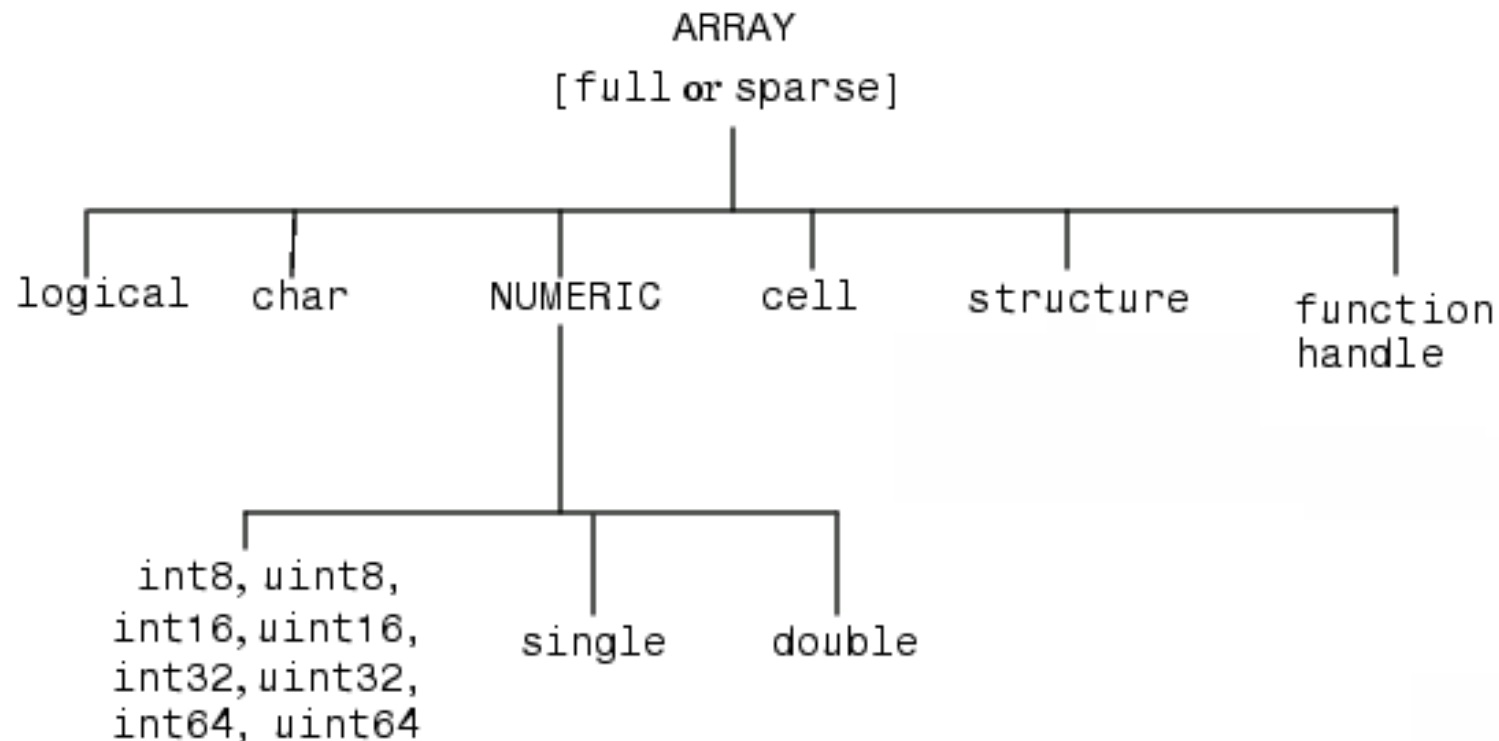
The instructions between catch and end are only executed in case of error among the first set of statements. One may use `lasterr` to get the error code that triggered the catch section.

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

All data types

- Matlab has 15 data types that can be used to build matrices or arrays



In addition there are user defined data types for object oriented programming: *user classes*, y *Java classes*

Identifying the type of data

- Description of the data type

```
>> type=class(x)
type =
double
>>
```

- Logical identification

```
isinteger(x)
isfloat(x)
ischar(x)
islogical(x)
iscell(x)
isstruct(x)
```

```
int8, uint8
int16, uint16
int32, uint32
int64, uint64
```

```
single
double
```

```
isempty([])
isinf(Inf)
isnan(NaN)
```

Character strings

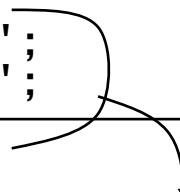
- In matlab strings are vectors of character (like in C)

```
>> str='Hello world';
>> whos
  Name      Size      Bytes  Class
  str      1x11      22    char array
Grand total is 11 elements using 22 bytes

>> str(7)
ans =
W

>> str=['H', 'o', 'l', 'a'];

>> nombres(1,:)= 'Rafael';
>> nombres(2,:)= 'Ana   ';
```



Variables that contain several names, build a matrix of char in which all the names have the same length. The conversion function `char` helps to build such matrix

```
>> names=char('Rafael', 'Ana');
```

Using **cell arrays** it is possible to store several strings of different lengths into one variables

Strings

- Other functions for strings
 - `strrep`: typical find-and-replace
`st2=strrep(st1,'find','replace');`
 - `findstr`: find a string within another
`pos = findstr('find', st);`
 - `strcat`: concat 2 or more strings
`text = strcat(st1, st2, st3);`
 - `sprintf`: builds a string. Equivalent to `sprintf` in C
`st=sprintf('I have %6.2f EUR',my_money);`

Structures and Cell Arrays

- Structures allows matlab to store several variables of different types under one variable name

```
>> dot.x=123;
>> dot.y=34;
>> dot.color='red';
>> dot
dot =
      x: 123
      y:  34
 color: 'red'
```

- Structures do not require previous definition
- Fields access is similar to C
- It is possible to use vectors of structures

```
>> punto(2).x=435;
```

Structures and Cell Arrays

- A cell array is used to build vectors in which each element may use a different data type:

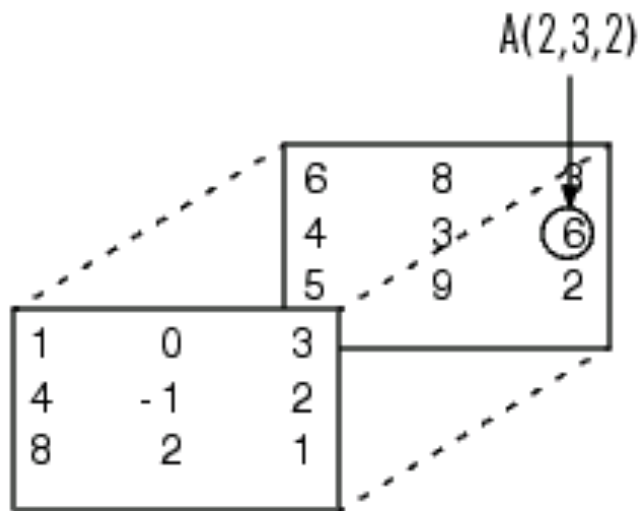
```
>> c={12,'Red',magic(4)};
>> c
c =
    [12]    'Red'    [4x4 double]

>> b{1}=12;
>> b{2}='Red';
>> b{3}=magic(4);
>> b
b =
    [12]    'Red'    [4x4 double]
```

- One should use { } instead of [] or ()
- The difference with structures is that you may use indexes instead of field names
- Structures and cell arrays are less efficient than matrices

Multi-dimensional matrix

- Matrices of more than 2 dimensions are called **Multidimensional Arrays**
- Matlab support all matrix operations in N dimensiones



$$A(:,:,1) =$$

1	0	3
4	-1	2
8	2	1

$$A(:,:,2) =$$

6	8	3
4	3	6
5	9	2

Multi-dimensional matrix

```
>> c=imread('autumn.tif');  
>> whos c  
Name      Size      Bytes  Class  
c         206x345x3 213210 uint8 array
```

Grand total is 213210 elements using 213210 bytes

```
>> imshow(c)
```

```
>> max(c(:))
```

```
ans =
```

```
248
```

todos los elementos

```
>> gris=(c(:,:,1)+c(:,:,2)+c(:,:,3))/3;
```

```
>> imshow(gris)
```



Try it yourself

- Everybody
 - declare variables of different data types
 - check the sizes of matrices
 - write a short function and call it from matlab
- Advanced
 - Load an image as a 3D matrix
 - Create a cell array

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Measuring time

- Basic functions to measure execution time
 - `tic` & `toc` measure elapsed time in seconds

```
>> tic; inv(inv(inv(randn(1000)))); toc  
Elapsed time is 10.015000 seconds.
```

```
tic  
    for k = 1:100  
        -- programa rápido --  
    end  
toc
```

- `cputime` measures CPU time in seconds

```
>> t=cputime; inv(inv(inv(randn(1000)))); e=cputime-t  
e =  
    9.5137
```

Code analysis

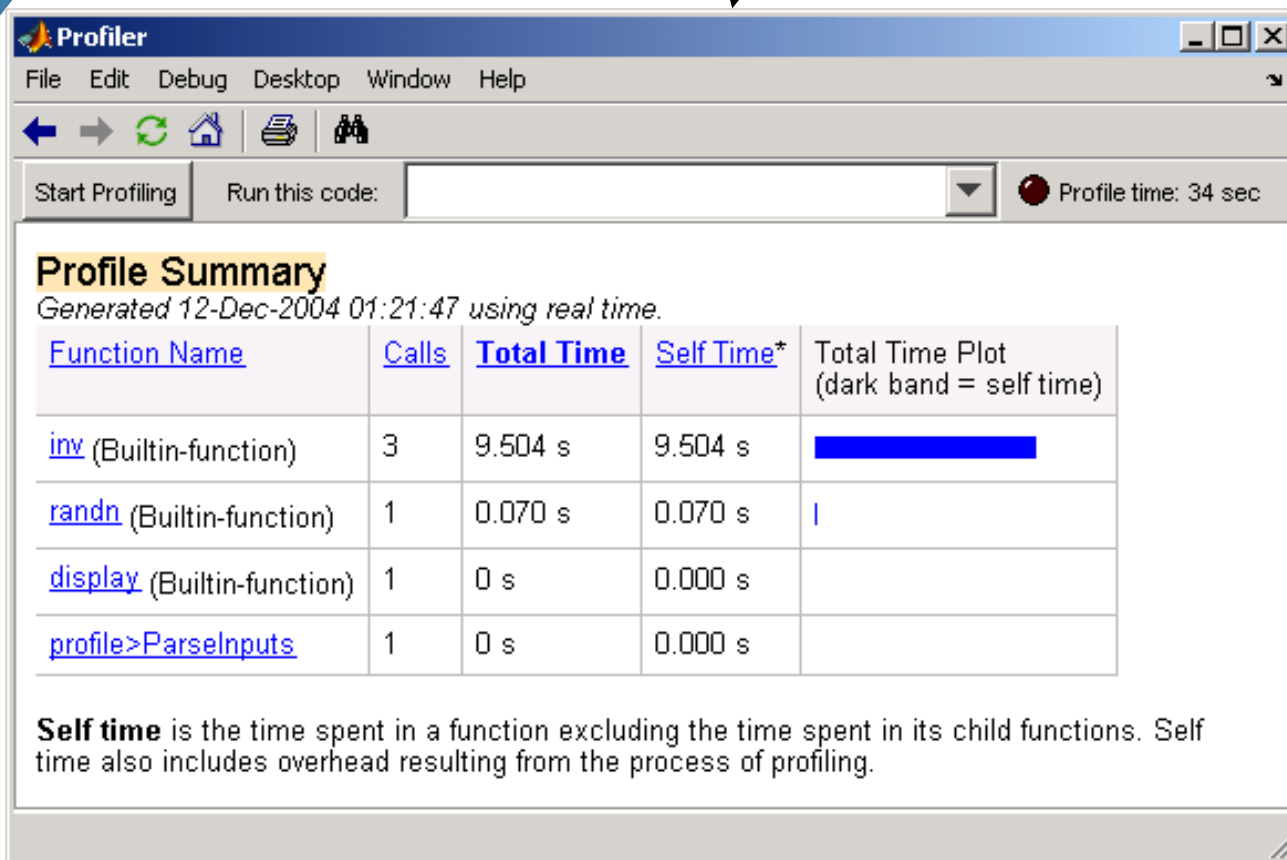
- profiler generates a report of the performance of the code

Graphical mode:

```
>> profile viewer
```

Commands

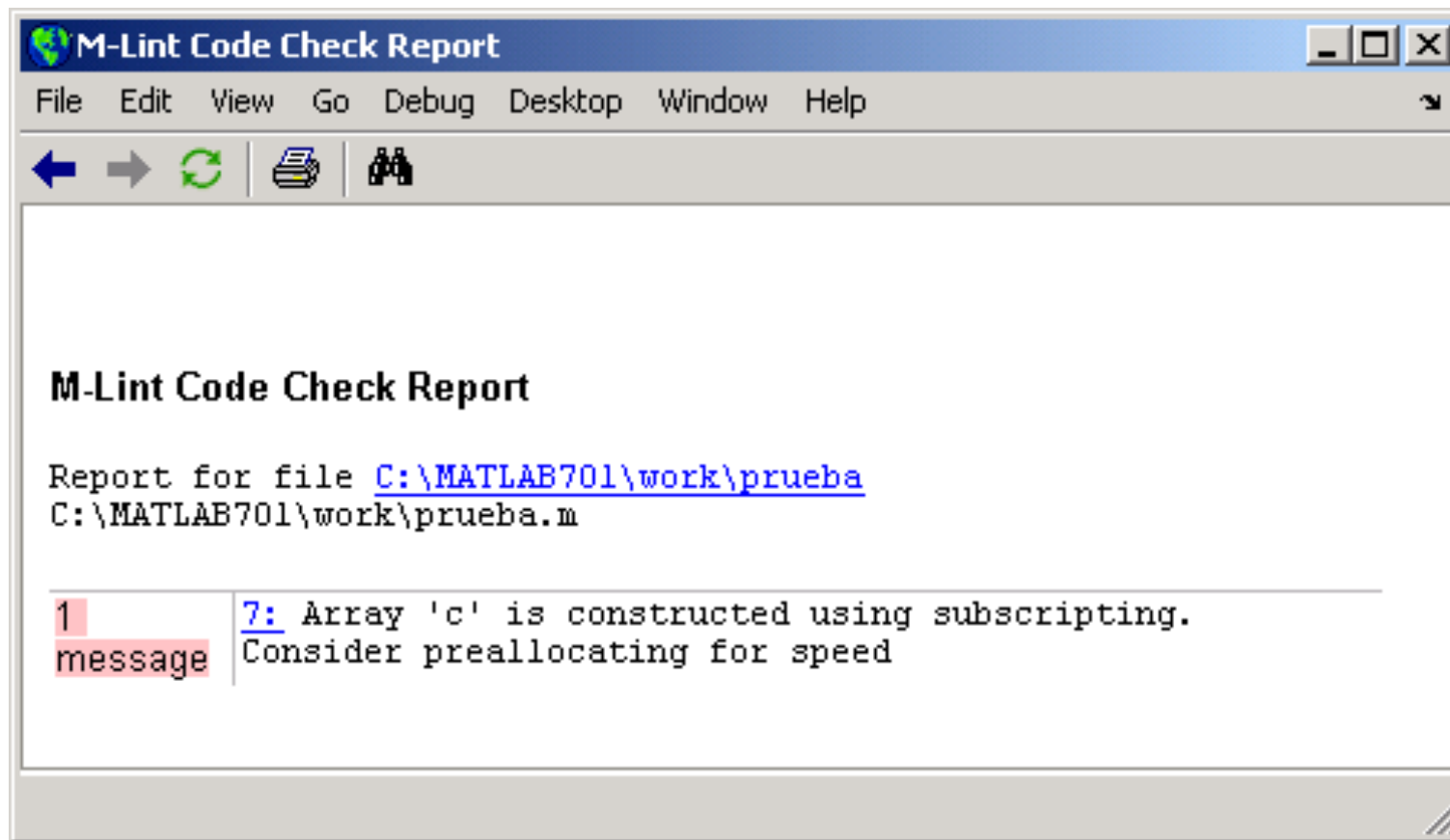
```
>> profile on  
>> inv(inv(inv(randn(1000))));  
>> profile off  
>> profile report
```



Profiler detects the function that it is worth improving.

Code analysis

- M-Lint is used to analyze the code and automatically detect possible improvements.



Try it yourself

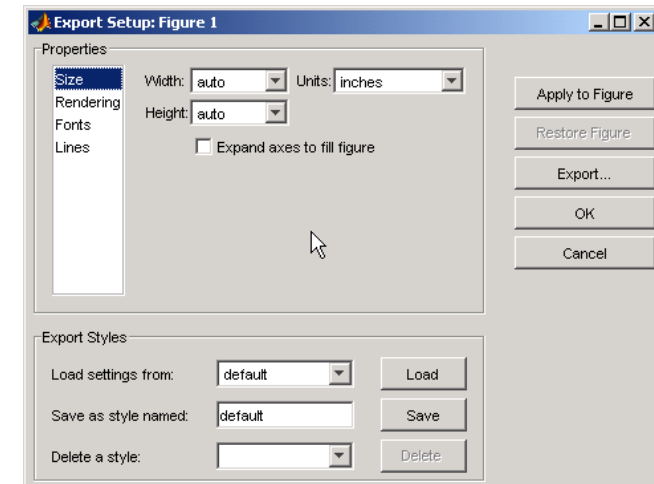
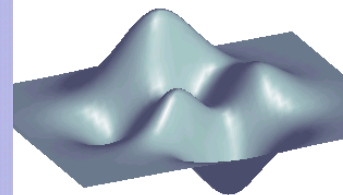
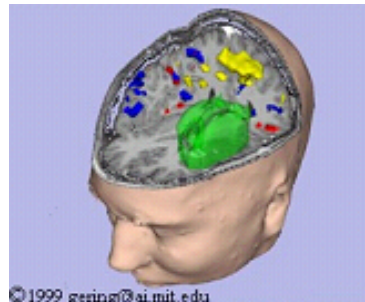
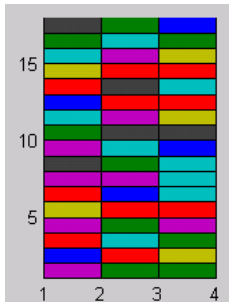
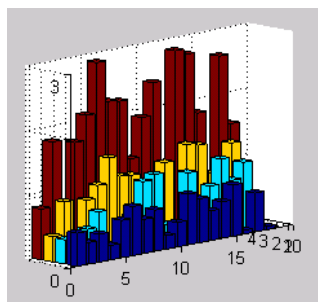
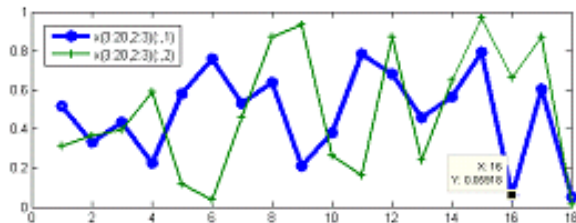
- Everybody
 - Test your function with M-Lint
 - Use tic and toc to measure execution time
- Advanced
 - Prepare a larger program in which one function calls another.
 - Run your program using profiler
 - Identify bottlenecks.

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

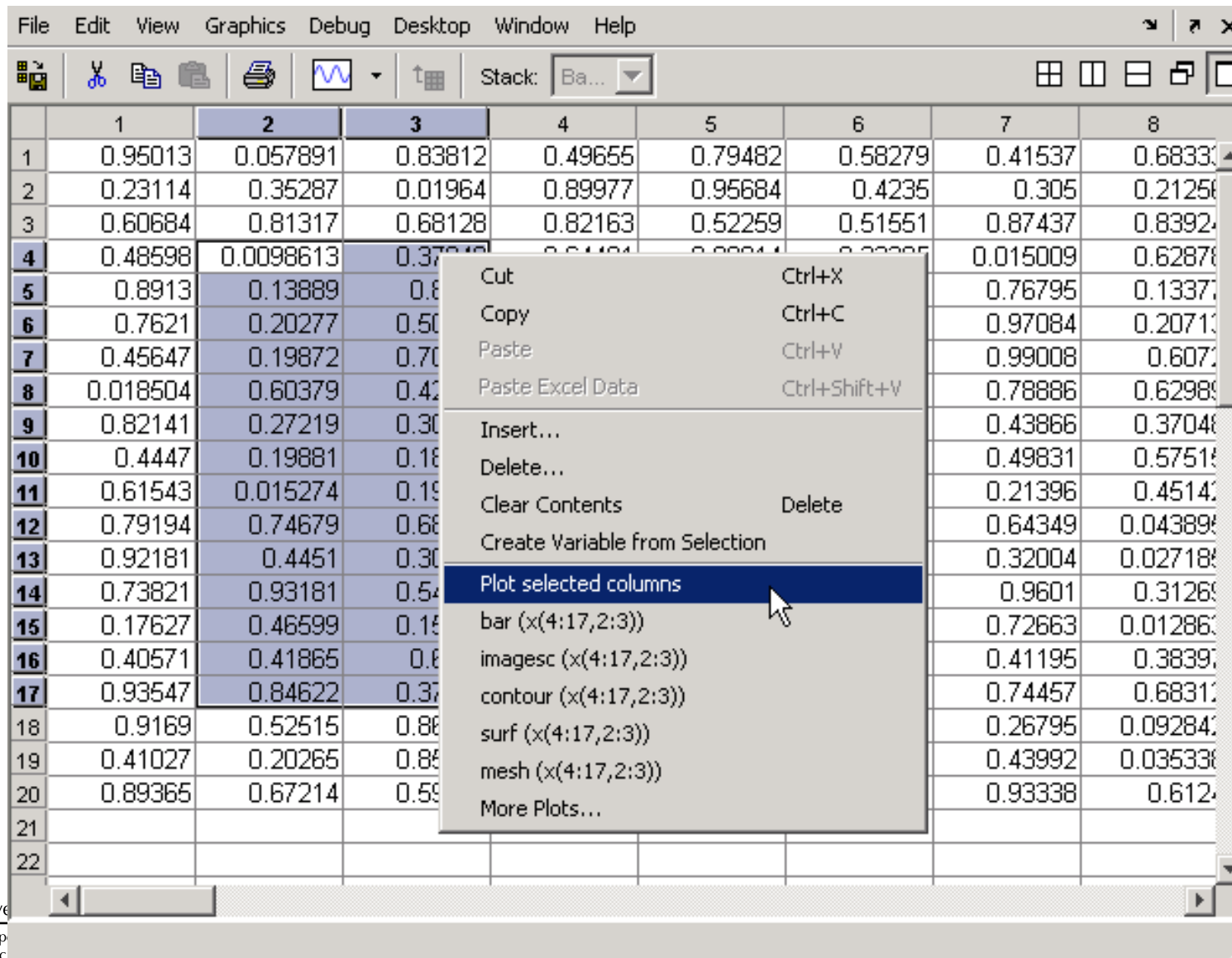
Create graphics

- Matlab can create different types of graphics:
 - to display data values
 - to create images/movies/VR/GIS
 - to generate a graphical use interface



Create graphics

- Create graphics directly from the matrix editor



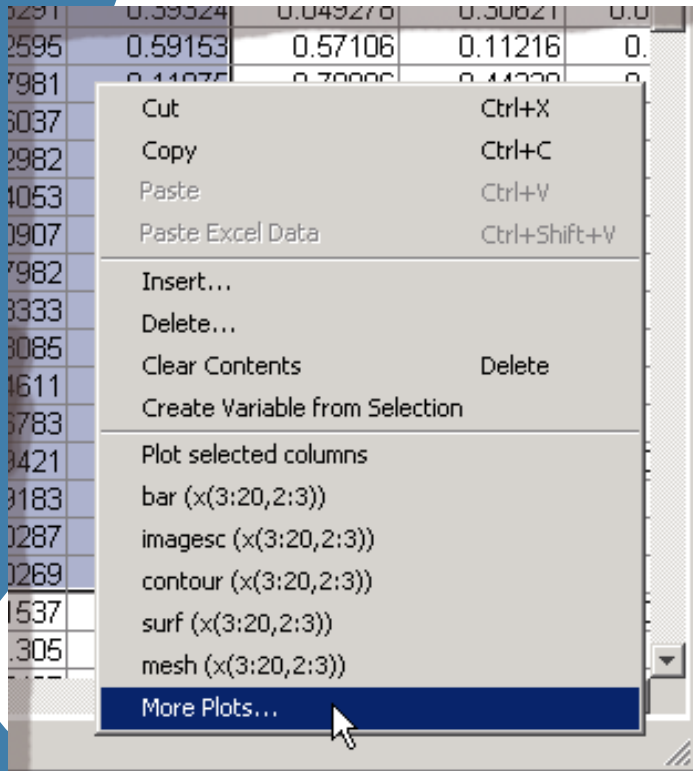
The screenshot shows the MATLAB Matrix Editor interface. The menu bar includes File, Edit, View, Graphics, Debug, Desktop, Window, and Help. The toolbar contains icons for matrix operations and a 'Stack' dropdown menu. The main area displays a matrix with columns 2, 3, and 4 selected. A context menu is open over the selected cells, listing various actions. The 'Plot selected columns' option is highlighted by the mouse cursor.

	1	2	3	4	5	6	7	8
1	0.95013	0.057891	0.83812	0.49655	0.79482	0.58279	0.41537	0.68330
2	0.23114	0.35287	0.01964	0.89977	0.95684	0.4235	0.305	0.21256
3	0.60684	0.81317	0.68128	0.82163	0.52259	0.51551	0.87437	0.83924
4	0.48598	0.0098613	0.37019	0.81484	0.99844	0.22285	0.015009	0.62876
5	0.8913	0.13889	0.8				0.76795	0.13370
6	0.7621	0.20277	0.50				0.97084	0.20710
7	0.45647	0.19872	0.70				0.99008	0.60720
8	0.018504	0.60379	0.42				0.78886	0.62989
9	0.82141	0.27219	0.30				0.43866	0.37046
10	0.4447	0.19881	0.18				0.49831	0.57519
11	0.61543	0.015274	0.19				0.21396	0.45140
12	0.79194	0.74679	0.68				0.64349	0.043899
13	0.92181	0.4451	0.30				0.32004	0.027189
14	0.73821	0.93181	0.54				0.9601	0.31269
15	0.17627	0.46599	0.18				0.72663	0.012860
16	0.40571	0.41865	0.8				0.41195	0.38390
17	0.93547	0.84622	0.37				0.74457	0.68310
18	0.9169	0.52515	0.86				0.26795	0.092840
19	0.41027	0.20265	0.88				0.43992	0.035338
20	0.89365	0.67214	0.59				0.93338	0.61200
21								
22								

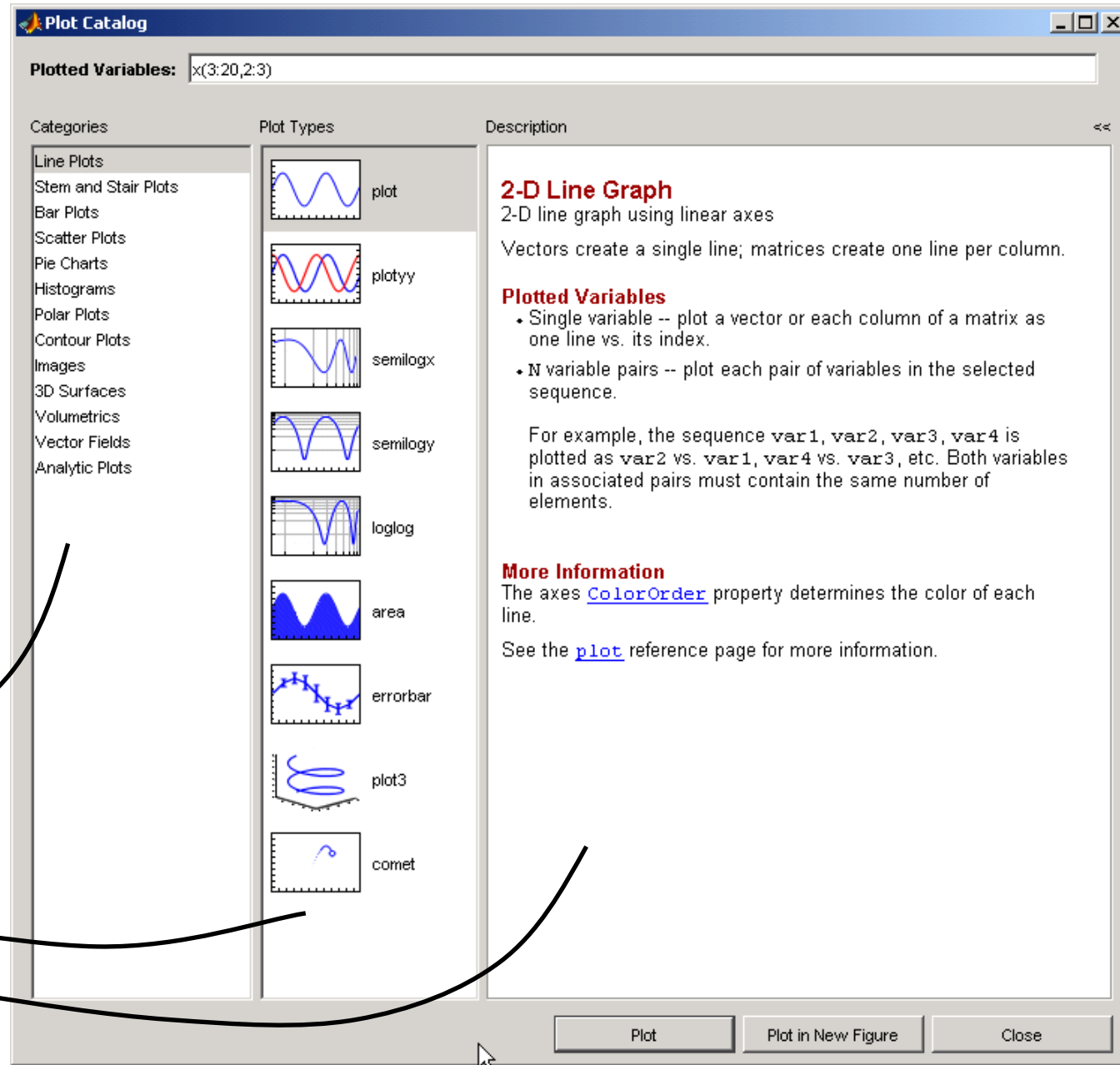
Context Menu Options:

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Paste Excel Data (Ctrl+Shift+V)
- Insert...
- Delete...
- Clear Contents (Delete)
- Create Variable from Selection
- Plot selected columns**
- bar (x(4:17,2:3))
- imagesc (x(4:17,2:3))
- contour (x(4:17,2:3))
- surf (x(4:17,2:3))
- mesh (x(4:17,2:3))
- More Plots...

Select graphics type



Matrix editor



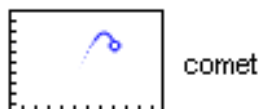
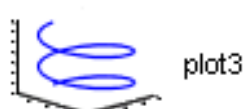
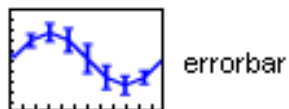
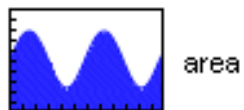
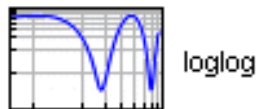
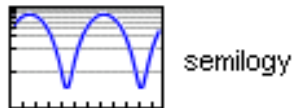
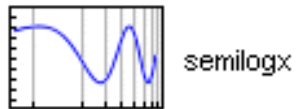
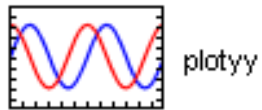
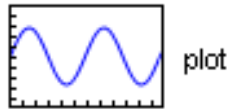
Categories

Plot types

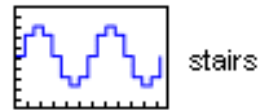
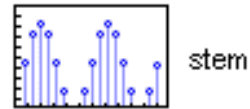
Description and function references

Types of graphs (1D, 2D)

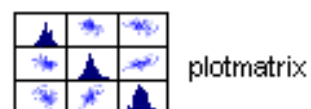
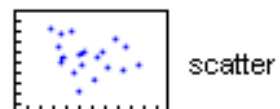
Line



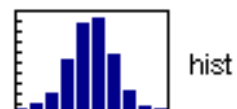
Stem & stair



Scatter



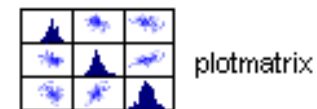
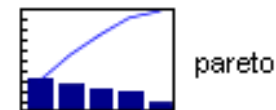
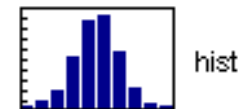
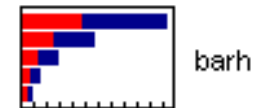
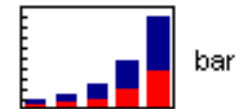
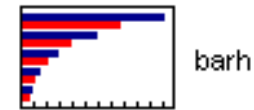
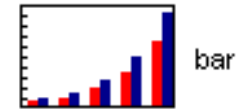
Histogram



Polar



Bar

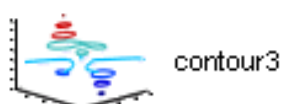
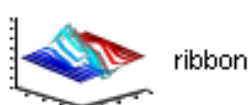
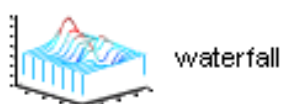
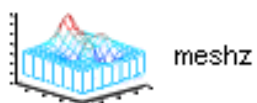
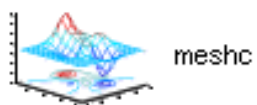
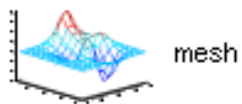
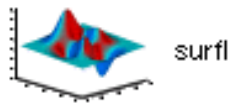
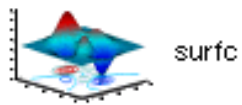
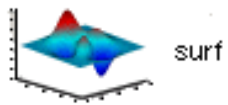


Pie

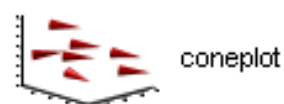
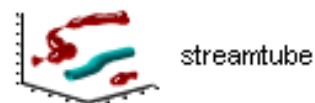
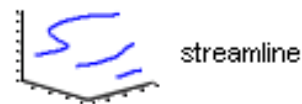
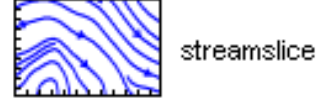
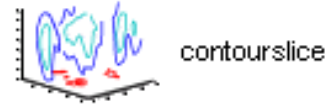
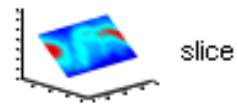


Types of graphs ($\geq 3D$)

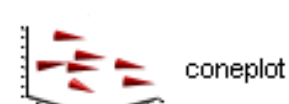
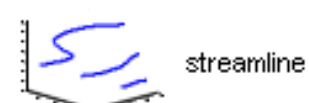
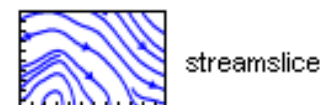
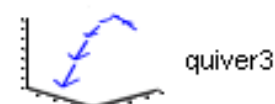
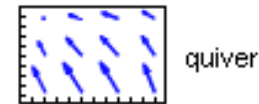
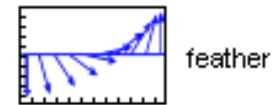
3D surfaces



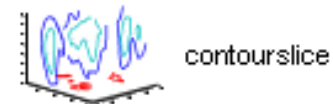
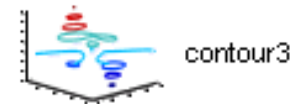
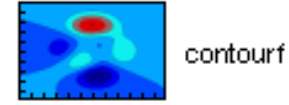
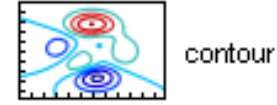
Volumetrics



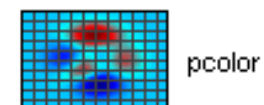
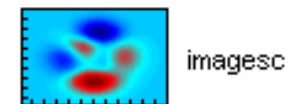
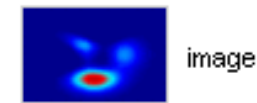
Vector Fields



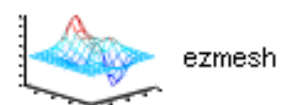
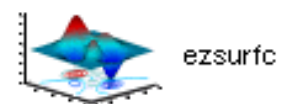
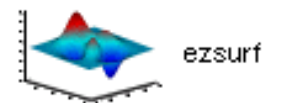
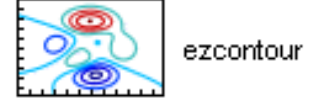
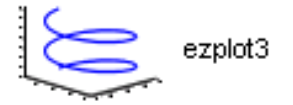
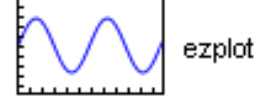
Contour



Images

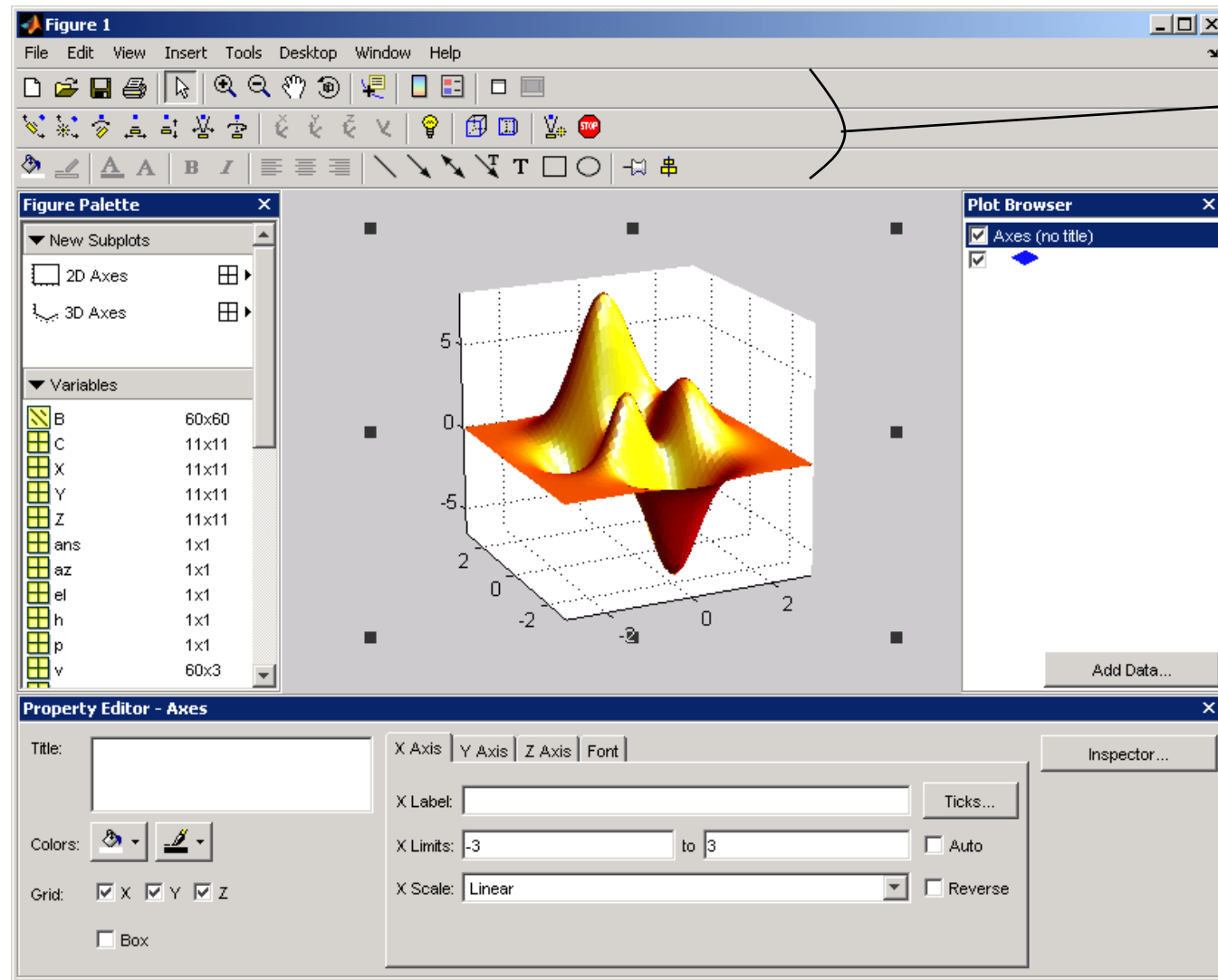


Analytic



Graph adjustments through menus

Ventana de la figura con todas las opciones activadas



→ Toolbars

Create new subplots

Variables of workspace

Object selector

Properties:

- Figure
- Axes
- Current Object

Save figures

- Using the menu

- File/Save As → .fig, .eps, .png, .jpeg, .bmp, .pcx, .tiff
- File/Generate M file

Esta opción nos permite ver qué comandos se utilizan para crear las modificaciones que hemos realizado por menú

- Using commands

- `hgsave my_fig` → `my_fig.fig` Can be loaded using `hgload`
- save the figure as an image
 - `print -depsc -tiff -r300 archivo`
 - `print -dpng -r150 archivo`

Contents

1. Introduction to Matlab.
2. Basic data types.
3. Matlab programming.
4. Advanced data types.
5. Code Optimization.
6. Graphical representation.
7. Developing standalone applications.

Compiler

- Converts Matlab code to C and generates an executable independent from Matlab
 - The program created does not require a Matlab license to work
 - It may execute faster (not always)
 - Compiler works with functions, not scripts

Installation (only the first time)

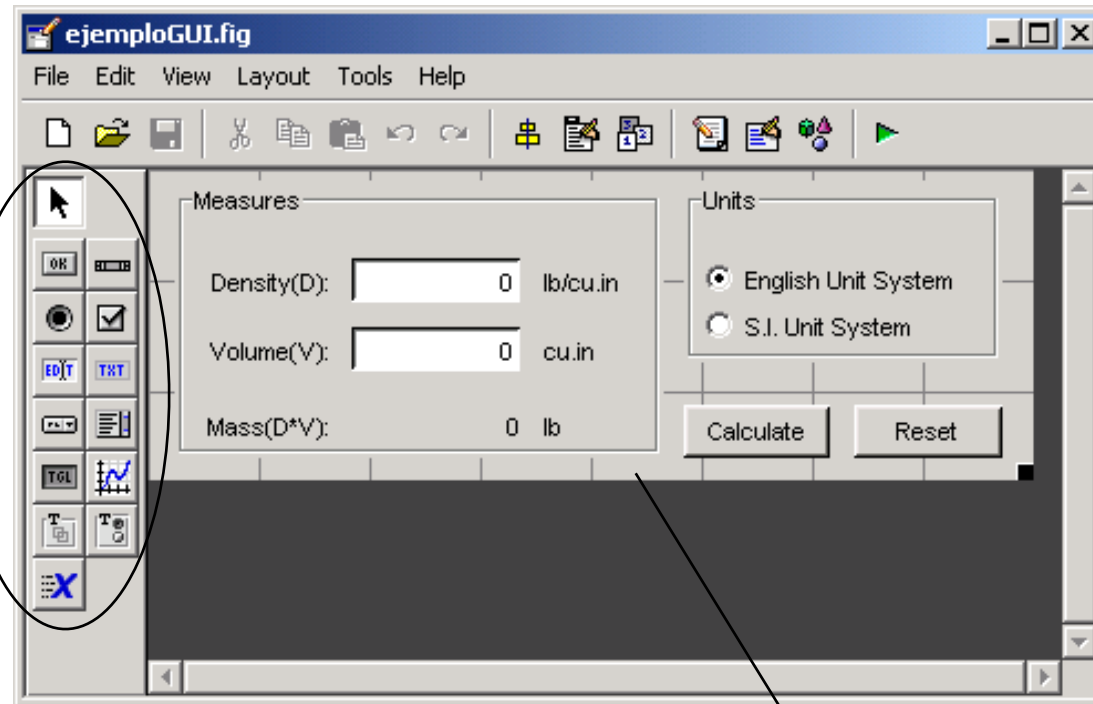
```
mbuild -setup
```

Command to compile

```
mcc -m prueba.m
```

Create graphical user interfaces

- Use guide from Matlab

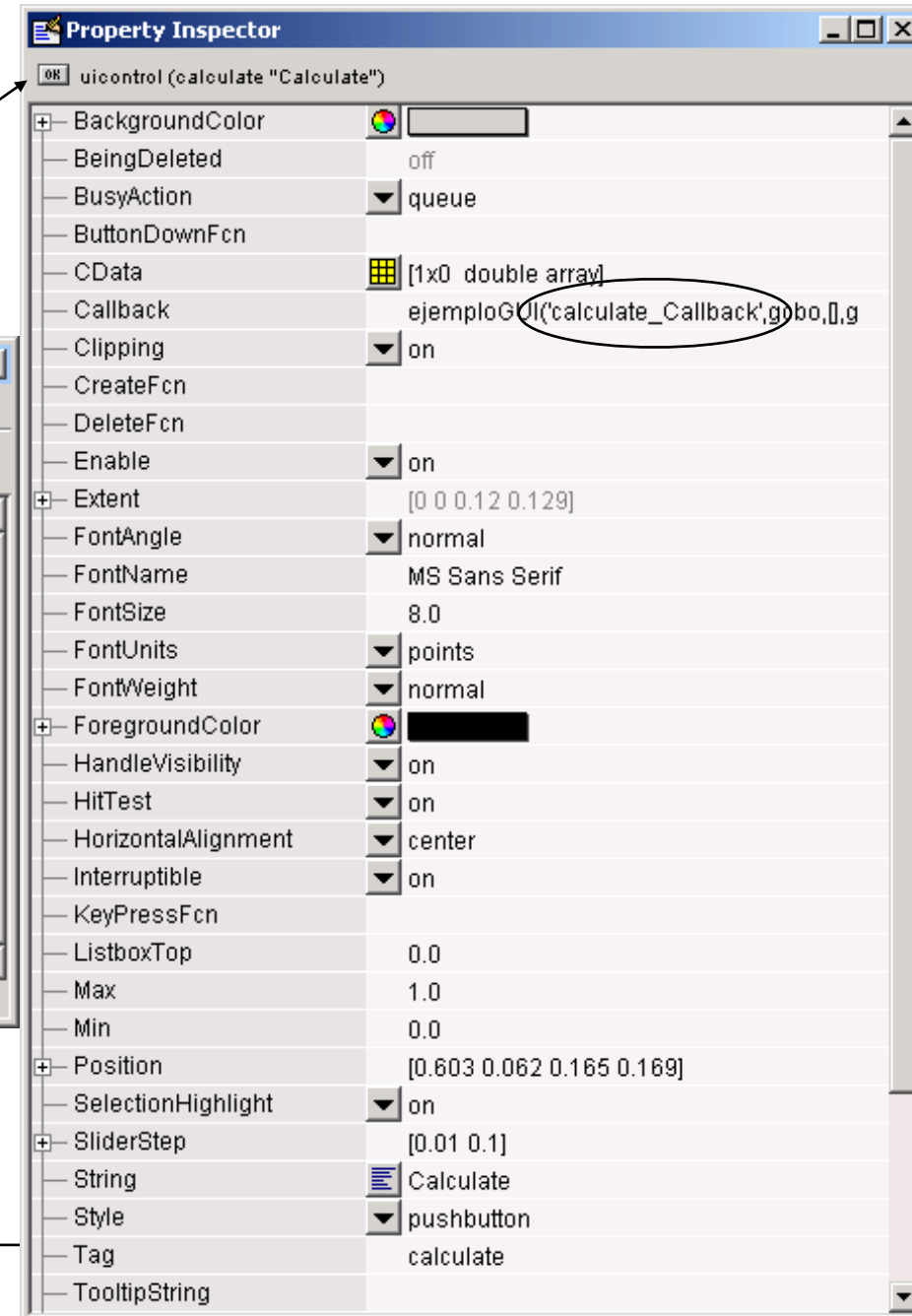
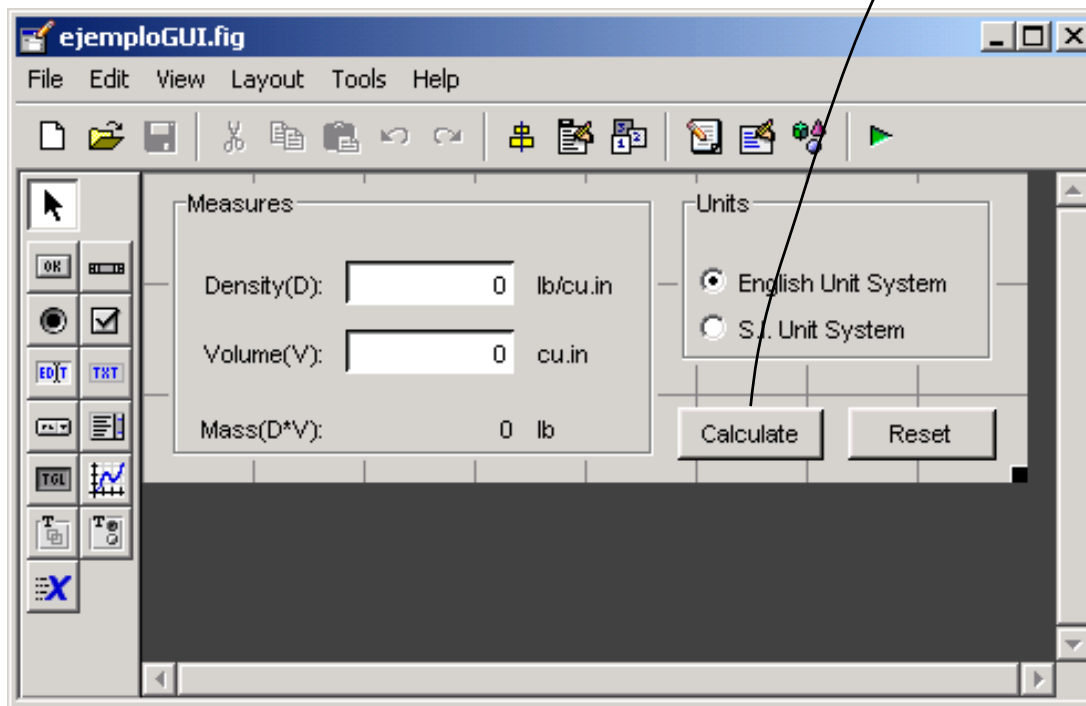


Object to draw

Application

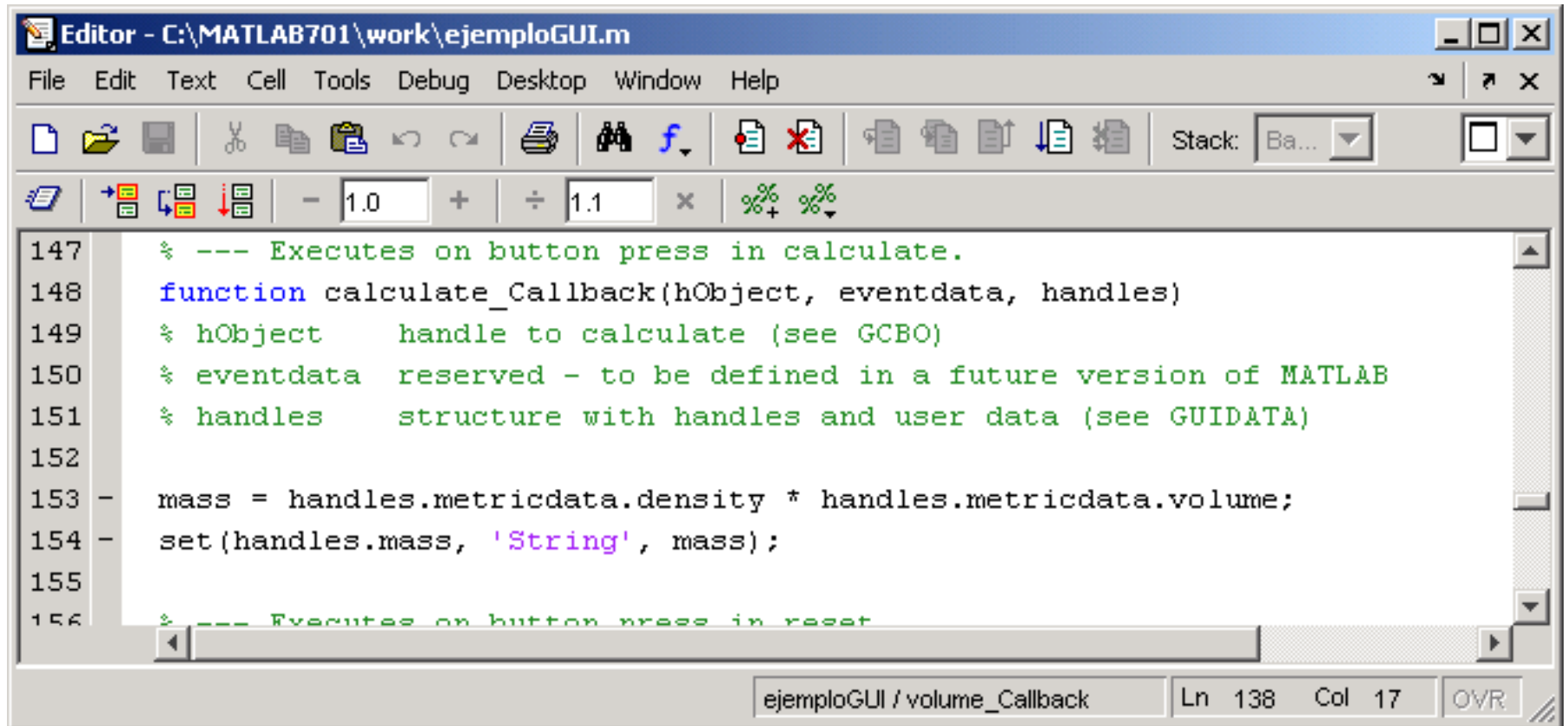
Create graphical user interfaces

- Each object has some attributes and a callback function



Create graphical user interfaces

- Guide generates a .m file ready to insert your application code



The screenshot shows a MATLAB Editor window titled "Editor - C:\MATLAB701\work\ejemploGUI.m". The window contains a MATLAB script with the following code:

```
147 % --- Executes on button press in calculate.
148 function calculate_Callback(hObject, eventdata, handles)
149 % hObject      handle to calculate (see GCBO)
150 % eventdata   reserved - to be defined in a future version of MATLAB
151 % handles     structure with handles and user data (see GUIDATA)
152
153 - mass = handles.metricdata.density * handles.metricdata.volume;
154 - set(handles.mass, 'String', mass);
155
156 % --- Executes on button press in reset
```

The status bar at the bottom of the window indicates the current position: "ejemploGUI / volume_Callback Ln 138 Col 17 OVR".

In general it is recommended to keep computing code isolated from interface functions

On-Line resources

- **Mathworks website**

 - <http://www.mathworks.com/support/>

 - Documentation
 - Support. Sorted by categories
 - Sample code
 - News
 - Software updates

- **Matlab Central**

 - Newsgroups
 - File Exchange
 - Link Exchange

- **Technical support by email**

 - You need to provide your active license code.
 - You must describe platform, operating system, etc.
 - Be very specific with your problem.

Try it yourself

- Everybody
 - Create a matrix
 - Create a graph
 - Edit graph properties
 - Save you graph, close Matlab, open your graph
- Advanced
 - Play with 3D surfaces
 - Add text to your graph
 - Use Guide to start an application